Cppcheck C/C++ static analyser

CONCEPTS

> By default only errors are shown. Add --enable=style (or all) to see warnings, style and performance findings.

Sources files or project
↓ preprocess
Configurations #ifdef variants
↓ analyse
Checks --enable ids
↓ report
Findings text / XML

> Cppcheck tries each #ifdef configuration itself, so missing -I paths and -D flags give fewer findings, not errors. Use --check-config to see what it could not resolve.

RUNNING

`cppcheck <file>`
Check one file

`cppcheck <dir>`
Check all sources in a folder, recursively

`cppcheck -j <n> <dir>`
Check with n threads

`cppcheck -q <dir>`
Quiet: print only findings

`cppcheck -i <path> <dir>`
Skip a file or folder

`cppcheck <dir> --file-filter=<pattern>`
Check only matching files

`cppcheck --language=c++ <file>`
Force C or C++ for all files

CHECKS AND LEVELS

`cppcheck --enable=warning <dir>`
Add warnings

`cppcheck --enable=style <dir>`
Add style findings

`cppcheck --enable=performance,portability <dir>`
Add several groups (comma list)

CHECKS AND LEVELS (cont.)

`cppcheck --enable=all <dir>`
Enable every group

`cppcheck --enable=unusedFunction <dir>`
Find unused functions

`cppcheck --enable=missingInclude <dir>`
Report unresolved includes

`cppcheck --check-level=exhaustive <dir>`
Deeper, slower analysis

`cppcheck --inconclusive <dir>`
Also report uncertain findings

`cppcheck --check-config <dir>`
Show missing includes and config problems

CONFIGURATION

`cppcheck -I <dir> <path>`
Add an include directory

`cppcheck -D<name> <path>`
Define a macro, check only that config

`cppcheck -U<name> <path>`
Treat a macro as undefined

`cppcheck --max-configs=<n> <path>`
Limit #ifdef configurations tried

`cppcheck --force <path>`
Check every configuration (can be slow)

`cppcheck --std=c++20 <path>`
Set the C++ standard

`cppcheck --std=c11 <file>`
Set the C standard

`cppcheck --platform=<name> <path>`
Set target types and sizes

`cppcheck --library=<name> <path>`
Load a library config (posix, qt, ...)

PROJECTS AND INCREMENTAL RUNS

`cppcheck --project=compile_commands.json`
Use a compilation database

PROJECTS AND INCREMENTAL RUNS (cont.)

`cppcheck --project=<file>.sln`
Import a Visual Studio solution

`cppcheck --project=<file>.cppcheck`
Use a GUI project file

`cppcheck --project=compile_commands.json -i<dir>`
Exclude a folder from a project

`cppcheck --cppcheck-build-dir=<dir> <path>`
Cache results, enable whole-program checks

OUTPUT

`cppcheck --template=gcc <path>`
GCC-style lines for editors

`cppcheck --template='{file}: {line}: {message}' <path>`
Custom line format

`cppcheck --xml <path> 2> <out.xml>`
XML report (written to stderr)

`cppcheck-htmlreport --file=<out.xml> --report-dir=<dir> --source-dir=.`
Turn XML into an HTML report

`cppcheck --error-exitcode=<n>`
Exit with n when findings exist

SUPPRESSIONS

`cppcheck --suppress=<id> <path>`
Suppress an id everywhere

`cppcheck --suppress=<id>:<file> <path>`
Suppress an id in one file

`cppcheck --suppressions-list=<file> <path>`
Read suppressions from a text file

`cppcheck --inline-suppr <path>`
Honour cppcheck-suppress comments

`// cppcheck-suppress <id>`
Suppress on the next or same line